Introdução ao Linux

Aula V: AWK

Prof. Dr. Marcelo Bianchi

Israel Dragone, Jamison Assunção
Leonardo Fabricius e Rafael Monteiro

26 de julho de 2018

A linguagem AWK

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

- $ awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}' dados.dat

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

- $ awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}' dados.dat

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

- $ awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}' dados.dat

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

- $ awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}' dados.dat

A linguagem AWK

AWK é uma linguagem de programação criada em 1977 por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan. O grande diferencial da linguagem é interpretar o arquivo linha por linha, executando o bloco de comandos principais a cada linha.

É amplamente aplicada em combinação com o Shell Script na manipulação e filtragem de dados, tratando o arquivo como uma sequência de registros, sendo cada linha um registro distinto onde cada palavra é um campo do registro.

Estrutura

- $ awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}' dados.dat
- $ cat dados.dat | awk 'BEGIN {print "Lendo o arquivo"} {print $0} END {print "fim da leitura"}

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

Como AWK interpreta esse arquivo?

O AWK possui algumas variáveis especiais para lidar com cada coluna ou então com a linha toda de uma vez.

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

Como AWK interpreta esse arquivo?

O AWK possui algumas variáveis especiais para lidar com cada coluna ou então com a linha toda de uma vez.

- **$0:** Guarda o conteúdo da linha toda;

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

Como AWK interpreta esse arquivo?

O AWK possui algumas variáveis especiais para lidar com cada coluna ou então com a linha toda de uma vez.

- **$0:** Guarda o conteúdo da linha toda;
- **$1:** Guarda o conteúdo da primeira coluna;

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

Como AWK interpreta esse arquivo?

O AWK possui algumas variáveis especiais para lidar com cada coluna ou então com a linha toda de uma vez.

- **$0:** Guarda o conteúdo da linha toda;
- **$1:** Guarda o conteúdo da primeira coluna;
- **$2:** Guarda o conteúdo da segunda coluna;

A linguagem AWK

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

Como AWK interpreta esse arquivo?

O AWK possui algumas variáveis especiais para lidar com cada coluna ou então com a linha toda de uma vez.

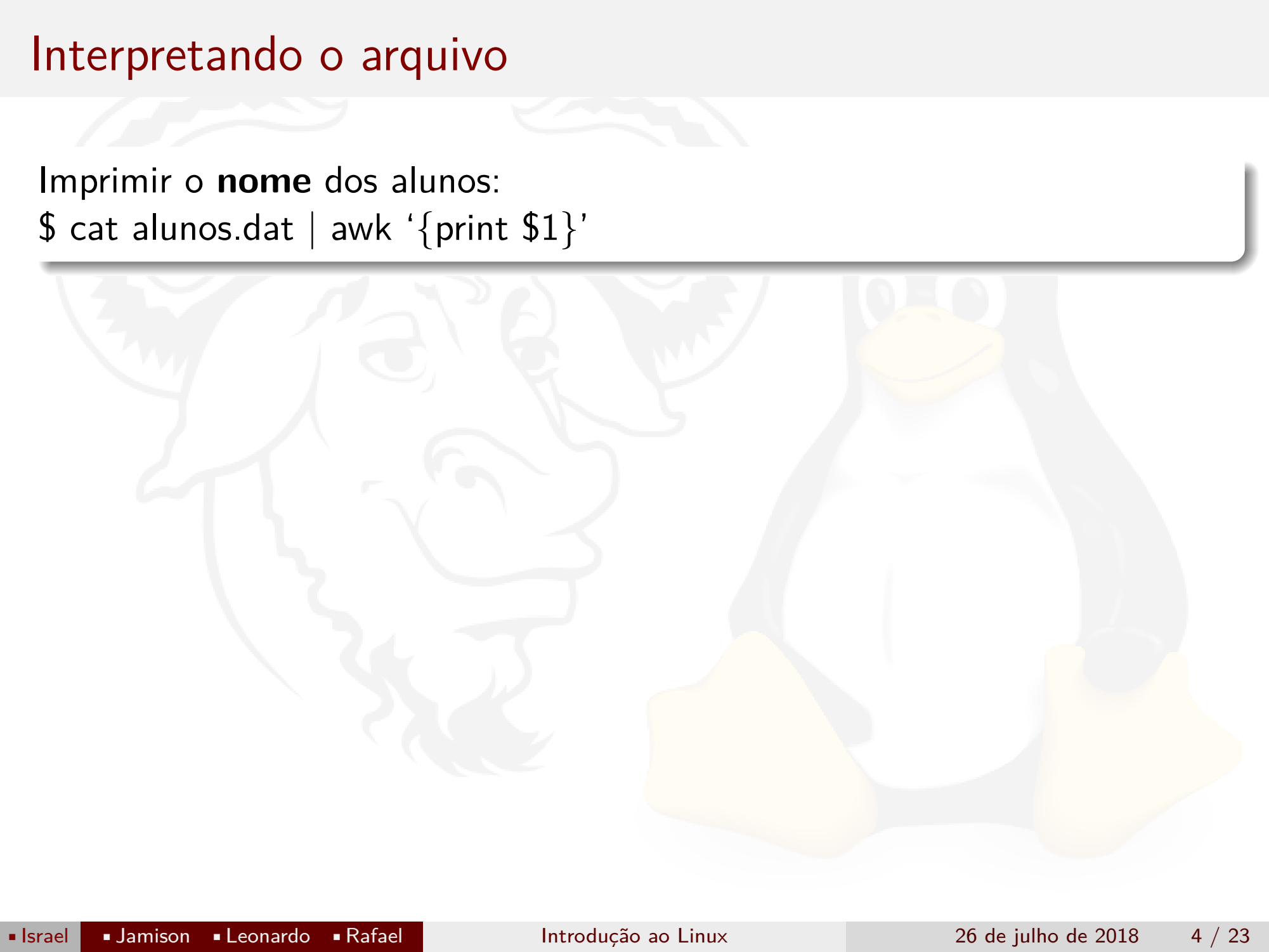
- **$0:** Guarda o conteúdo da linha toda;
- **$1:** Guarda o conteúdo da primeira coluna;
- **$2:** Guarda o conteúdo da segunda coluna;
- **$n:** Guarda o contúedo da enéseima coluna.

Interpretando o arquivo

Interpretando o arquivo

Imprimir o **nome** dos alunos:

$ cat alunos.dat | awk '{print $1}'



Interpretando o arquivo

Imprimir o **nome** dos alunos:
$ cat alunos.dat | awk '{print $1}'

```
Alunos
João
Pedro
Arthur
Anita
```

Interpretando o arquivo

Imprimir o **nome** dos alunos:
$ cat alunos.dat | awk '{print $1}'

```
Alunos
João
Pedro
Arthur
Anita
```

Imprimir o **curso**:
$ cat alunos.dat | awk '{print $4}'

Interpretando o arquivo

Imprimir o **nome** dos alunos:
$ cat alunos.dat | awk '{print $1}'

```
Alunos
João
Pedro
Arthur
Anita
```

Imprimir o **curso**:
$ cat alunos.dat | awk '{print $4}'

```
Curso
Computação
Música
Naval
Gerontologia
```

Filtrando a saída

Filtrando a saída

Mais algumas variáveis especiais

Filtrando a saída

Mais algumas variáveis especiais

- **NR:** Guarda o número da linha atual;

Filtrando a saída

Mais algumas variáveis especiais

- **NR:** Guarda o número da linha atual;
- **NF:** Guarda o número de colunas da linha.

Filtrando a saída

Mais algumas variáveis especiais

- **NR:** Guarda o número da linha atual;
- **NF:** Guarda o número de colunas da linha.

Removendo o cabeçalho do arquivo

Imprimir o **nome** dos alunos :

$ cat alunos.dat | awk 'NR>1 {print $1}'

Filtrando a saída

Mais algumas variáveis especiais

- **NR:** Guarda o número da linha atual;
- **NF:** Guarda o número de colunas da linha.

Removendo o cabeçalho do arquivo

Imprimir o **nome** dos alunos :

```
$ cat alunos.dat | awk 'NR>1 {print $1}'
```

```
João
Pedro
Arthur
Anita
```

Filtrando a saída

Filtrando a saída

Imprimir **nome** e **idade** sem o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $1, $2}'
```

Filtrando a saída

Imprimir **nome** e **idade** sem o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $1, $2}'
```

```
João 19
Pedro 25
Arthur 22
Anita 18
```

Filtrando a saída

Imprimir **nome** e **idade** sem o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $1, $2}'
```

```
João 19
Pedro 25
Arthur 22
Anita 18
```

Imprimir todo o arquivo exceto o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $0}'
```

Filtrando a saída

Imprimir **nome** e **idade** sem o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $1, $2}'
```

```
João 19
Pedro 25
Arthur 22
Anita 18
```

Imprimir todo o arquivo exceto o cabeçalho

```
$ cat alunos.dat | awk 'NR>1 {print $0}'
```

```
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
```

Filtrando a saída

Filtrando a saída

Quando usar o NF?

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Marcela 20 FFLCH 6
Pedro 25 ECA Música 12
Julia IO 5
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Felipe 23 FAU Arquitetura
Anita 18 EACH Gerontologia 6

Quando usar o NF?

Quando usar o NF?

Visualizar os alunos com dados completos

```
$ cat alunos.dat | awk 'NR>1 && NF==5 {print $0}'
```

Quando usar o NF?

Visualizar os alunos com dados completos

```
$ cat alunos.dat | awk 'NR>1 && NF==5 {print $0}'
```

João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Anita 18 EACH Gerontologia 6

Quando usar o NF?

Visualizar os alunos com dados completos

```
$ cat alunos.dat | awk 'NR>1 && NF==5 {print $0}'
```

João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Anita 18 EACH Gerontologia 6

Visualizar o **nome** dos alunos com dados incompletos

```
$cat alunos.dat | awk 'NR>1 && NF<5 {print $1}'
```

Quando usar o NF?

Visualizar os alunos com dados completos

```
$ cat alunos.dat | awk 'NR>1 && NF==5 {print $0}'
```

João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Anita 18 EACH Gerontologia 6

Visualizar o **nome** dos alunos com dados incompletos

```
$cat alunos.dat | awk 'NR>1 && NF<5 {print $1}'
```

Marcela
Julia
Felipe

Média de idade da turma

Média de idade da turma

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Marcela 20 FFLCH 6
Pedro 25 ECA Música 12
Julia IO 5
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Felipe 23 FAU Arquitetura
Anita 18 EACH Gerontologia 6

Média de idade da turma

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Marcela 20 FFLCH 6
Pedro 25 ECA Música 12
Julia IO 5
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Felipe 23 FAU Arquitetura
Anita 18 EACH Gerontologia 6

```
$ cat alunos.dat | awk 'BEGIN{media=0;i=0} (NR>1 && NF==5)
{media+=$2,i++} END{print "A média de idade da turma é de media/i,
contando somente os alunos com dados completos"}
```

Média de idade da turma

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Marcela 20 FFLCH 6
Pedro 25 ECA Música 12
Julia IO 5
Arthur 22 POLI Naval 6
Bruna 24 IF Física 7
Felipe 23 FAU Arquitetura
Anita 18 EACH Gerontologia 6

$ cat alunos.dat | awk 'BEGIN{media=0;i=0} (NR>1 && NF==5) {media+=$2,i++} END{print "A média de idade da turma é de media/i, contando somente os alunos com dados completos"}

A média de idade da turma é de 21.6 anos, contando somente os alunos com dados completos

Operações matemáticas básicas

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

- **Igual:** $1==$2, $1==X, $1=="palavra"

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

- **Igual:** $1==$2, $1==X, $1=="palavra"
- **Diferente:** $1!=$2, $1!=X, $1!="palavra"

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

- **Igual:** $1==$2, $1==X, $1=="palavra"
- **Diferente:** $1!=$2, $1!=X, $1!="palavra"
- **Maior e menor:** $1>$2, $1<X;

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

- **Igual:** $1==$2, $1==X, $1=="palavra"
- **Diferente:** $1!=$2, $1!=X, $1!="palavra"
- **Maior e menor:** $1>$2, $1<X;
- **Maior ou igual e menor ou igual:** $1>=$2, $1<=X

Operações matemáticas básicas

- **Soma:** A+B, $1+B, $1+$2
- **Subtração:** A-B, $1-B, $1-$2
- **Multiplicação:** A*B, $1*B, $1*$2
- **Divisão:** A/B, $1/B, $1/$2;
- **Resto da divisão:** A%B, $1%B, $1%$2
- **Potenciação:** $1^B, $1**$2

Comparações

- **Igual:** $1==$2, $1==X, $1=="palavra"
- **Diferente:** $1!=$2, $1!=X, $1!="palavra"
- **Maior e menor:** $1>$2, $1<X;
- **Maior ou igual e menor ou igual:** $1>=$2, $1<=X
- **Operadores lógicos:** &&, || e !

Estruturando as comparações

Estruturando as comparações

- **maior que:**
  $ awk '$1 > 3 {print $0}' file

Estruturando as comparações

- **maior que:**
  $ awk '$1 > 3 {print $0}' file
- **Entre uma faixa de valores**:
  $ awk '($1 < 3) && ($1 > 0) {print $0}' file

Estruturando as comparações

- **maior que:**
  $ awk '$1 > 3 {print $0}' file
- **Entre uma faixa de valores**:
  $ awk '($1 < 3) && ($1 > 0) {print $0}' file
- **OU:**
  $ awk '($1/$2 == $3) || ($1/$2 == $3-1) {print $0}' file

Estruturando as comparações

- **maior que:**
  $ awk '$1 > 3 {print $0}' file
- **Entre uma faixa de valores**:
  $ awk '($1 < 3) && ($1 > 0) {print $0}' file
- **OU:**
  $ awk '($1/$2 == $3) || ($1/$2 == $3-1) {print $0}' file
- **E e OU:**
  $ awk '($3==1 && $1 > 2) || ($2 == 1) {print $0}' file

Estruturando as comparações

- **maior que:**
  $ awk '$1 > 3 {print $0}' file
- **Entre uma faixa de valores**:
  $ awk '($1 < 3) && ($1 > 0) {print $0}' file
- **OU:**
  $ awk '($1/$2 == $3) || ($1/$2 == $3-1) {print $0}' file
- **E e OU:**
  $ awk '($3==1 && $1 > 2) || ($2 == 1) {print $0}' file
- **NOT:**
  $ awk '!($7**2>100) {print $0}' file

Exercícios

Atividade premier_league.dat

Exercícios

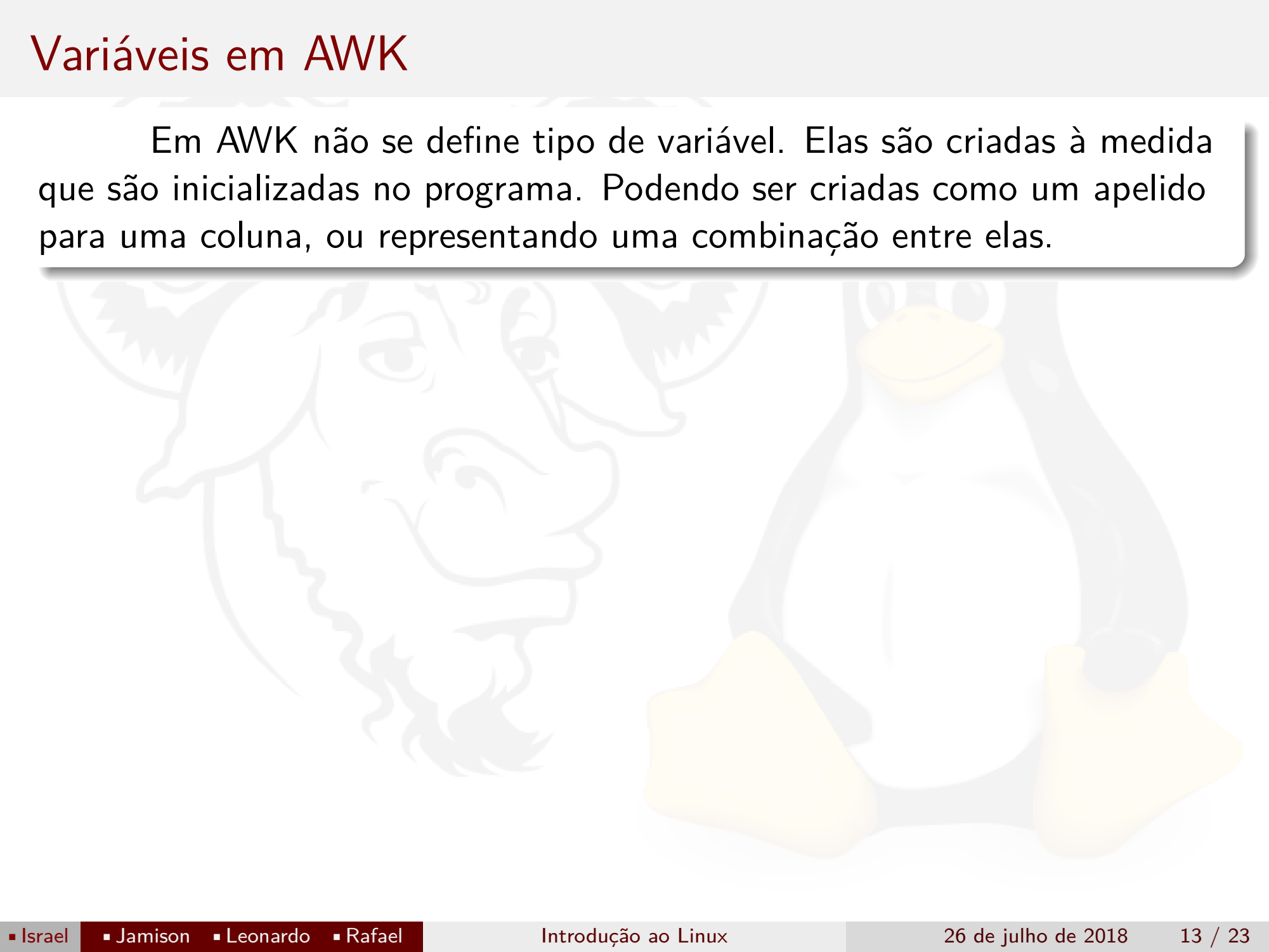
Atividade premier_league.dat

1. Exiba o nome dos clubes, gols marcados e gols sofridos;
2. Exiba o número de pontos e o nome dos clubes que sofreram menos de um gol por jogo;
3. Encontre os clubes que obtiveram aproveitamento entre 40 e 60%;
4. Calcule a média de gols marcados por clube no campeonato;
5. Exiba os clubes que tiveram mais que 15 vitórias e menos que 10 derrotas.
6. Use o awk para exibir a classificação, número de pontos, aproveitamento e o nome do time nessa ordem;
7. Calcule a média de gols marcados e sofridos por cada time;
8. Calcule a média de pontos e de aproveitamento dos clubes no campeonato;

Variáveis em AWK

Variáveis em AWK

Em AWK não se define tipo de variável. Elas são criadas à medida que são inicializadas no programa. Podendo ser criadas como um apelido para uma coluna, ou representando uma combinação entre elas.



Variáveis em AWK

Em AWK não se define tipo de variável. Elas são criadas à medida que são inicializadas no programa. Podendo ser criadas como um apelido para uma coluna, ou representando uma combinação entre elas.

```
Nome E P1 P2 P3 T
Ana 8.5 9.4 8.0 6.9 8.9
Carlos 10.0 9.4 8.7 9.8 9.5
Zilda 9.0 8.9 8.7 9.5 9.9
```

Variáveis em AWK

Em AWK não se define tipo de variável. Elas são criadas à medida que são inicializadas no programa. Podendo ser criadas como um apelido para uma coluna, ou representando uma combinação entre elas.

Nome E P1 P2 P3 T
Ana 8.5 9.4 8.0 6.9 8.9
Carlos 10.0 9.4 8.7 9.8 9.5
Zilda 9.0 8.9 8.7 9.5 9.9

$$média = \frac{2 \cdot E + 5 \cdot \frac{(P1+P2+P3)}{3} + 3 \cdot T}{10}$$

Variáveis em AWK

Em AWK não se define tipo de variável. Elas são criadas à medida que são inicializadas no programa. Podendo ser criadas como um apelido para uma coluna, ou representando uma combinação entre elas.

Nome E P1 P2 P3 T
Ana 8.5 9.4 8.0 6.9 8.9
Carlos 10.0 9.4 8.7 9.8 9.5
Zilda 9.0 8.9 8.7 9.5 9.9

$$média = \frac{2 \cdot E + 5 \cdot \frac{(P1+P2+P3)}{3} + 3 \cdot T}{10}$$

- $ cat notas.dat | awk '{nome=$1; media=($2*0.2 + (($3+$4+$5)/3)*0.5 + $6*0.3); print nome, media}'

Opções importantes - Separadores de coluna

Por padrão, o AWK espera que as colunas do arquivo de entrada estejam separas por espaço. Mas é muito comum encontrar arquivos com diversos outros separadores, pricipalmente vírgulas e pipes (|). Para lidar com esses arquivos se usa a opção -**F** especificando o separador de colunas.

Opções importantes - Separadores de coluna

Por padrão, o AWK espera que as colunas do arquivo de entrada estejam separas por espaço. Mas é muito comum encontrar arquivos com diversos outros separadores, pricipalmente vírgulas e pipes (|). Para lidar com esses arquivos se usa a opção -**F** especificando o separador de colunas.

```
Nome,E,P1,P2,P3,T,Media
Ana,8.5,9.4,8.0,6.9,8.9,8.4
Carlos,10.0,9.4,8.7,9.8,9.5,9.5
Zilda,9.0,8.9,8.7,9.5,9.9,9.3
```

Opções importantes - Separadores de coluna

Por padrão, o AWK espera que as colunas do arquivo de entrada estejam separas por espaço. Mas é muito comum encontrar arquivos com diversos outros separadores, pricipalmente vírgulas e pipes (|). Para lidar com esses arquivos se usa a opção -**F** especificando o separador de colunas.

```
Nome,E,P1,P2,P3,T,Media
Ana,8.5,9.4,8.0,6.9,8.9,8.4
Carlos,10.0,9.4,8.7,9.8,9.5,9.5
Zilda,9.0,8.9,8.7,9.5,9.9,9.3
```

```
cat notas.dat | awk -F',' 'NR>1 {print $1,$7}' ;
```

Opções importantes - Separadores de coluna

Por padrão, o AWK espera que as colunas do arquivo de entrada estejam separas por espaço. Mas é muito comum encontrar arquivos com diversos outros separadores, pricipalmente vírgulas e pipes (|). Para lidar com esses arquivos se usa a opção -**F** especificando o separador de colunas.

```
Nome,E,P1,P2,P3,T,Media
Ana,8.5,9.4,8.0,6.9,8.9,8.4
Carlos,10.0,9.4,8.7,9.8,9.5,9.5
Zilda,9.0,8.9,8.7,9.5,9.9,9.3
```

```
cat notas.dat | awk -F',' 'NR>1 {print $1,$7}' ;
```

```
Ana 8.4
Carlos 9.5
Zilda 9.3
```

Opções importantes - Separadores de coluna

```
Nome|E,P1,P2,P3,T|Media
Ana|8.5,9.4,8.0,6.9,8.9|8.4
Carlos|10.0,9.4,8.7,9.8,9.5|9.5
Zilda|9.0,8.9,8.7,9.5,9.9|9.3
```

Opções importantes - Separadores de coluna

```
Nome|E,P1,P2,P3,T|Media
Ana|8.5,9.4,8.0,6.9,8.9|8.4
Carlos|10.0,9.4,8.7,9.8,9.5|9.5
Zilda|9.0,8.9,8.7,9.5,9.9|9.3
```

```
$ cat notas.dat | awk -F'[,|]' 'NR>1 {print $1,$7}'
```

Opções importantes - Separadores de coluna

```
Nome|E,P1,P2,P3,T|Media
Ana|8.5,9.4,8.0,6.9,8.9|8.4
Carlos|10.0,9.4,8.7,9.8,9.5|9.5
Zilda|9.0,8.9,8.7,9.5,9.9|9.3
```

```
$ cat notas.dat | awk -F'[,|]' 'NR>1 {print $1,$7}'
```

```
Ana 8.4
Carlos 9.5
Zilda 9.3
```

Opções importantes

Procurar por uma expressão

Opções importantes

Procurar por uma expressão

- $ **cat arquivo.dat** | **awk '$0 ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome;

Opções importantes

Procurar por uma expressão

- $ **cat arquivo.dat** | **awk '$0 ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome;
- $ **cat arquivo.dat** | **awk '$col ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome na coluna **col**;

Opções importantes

Procurar por uma expressão

- $ **cat arquivo.dat | awk '$0 ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome;
- $ **cat arquivo.dat | awk '$col ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome na coluna **col**;
- $ **cat arquivo.dat | awk '$col ~ /^car/ {print $1, $3}'** $\longrightarrow$ Imprime as colunas 1 e 3 de todas as linhas onde encontrar a sequência de caracteres que se **inicia** com **car** na coluna **col**;

Opções importantes

Procurar por uma expressão

- $ **cat arquivo.dat | awk '$0 ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome;
- $ **cat arquivo.dat | awk '$col ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome na coluna **col**;
- $ **cat arquivo.dat | awk '$col ~ /^car/ {print $1, $3}'** $\longrightarrow$ Imprime as colunas 1 e 3 de todas as linhas onde encontrar a sequência de caracteres que se **inicia** com **car** na coluna **col**;
- $ **cat arquivo.dat | awk '$col ~ /car$/ {print $2, $3}'** $\longrightarrow$ Imprime as colunas 2 e 3 de todas as linhas onde encontrar a sequência de caracteres que **termina** em **car** na coluna **col**;

Opções importantes

Procurar por uma expressão

- $ **cat arquivo.dat | awk '$0 ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome;
- $ **cat arquivo.dat | awk '$col ~ /nome/'** $\longrightarrow$ Imprime todas as linhas onde encontrar a sequência de caracteres nome na coluna **col**;
- $ **cat arquivo.dat | awk '$col ~ /^car/ {print $1, $3}'** $\longrightarrow$ Imprime as colunas 1 e 3 de todas as linhas onde encontrar a sequência de caracteres que se **inicia** com **car** na coluna **col**;
- $ **cat arquivo.dat | awk '$col ~ /car$/ {print $2, $3}'** $\longrightarrow$ Imprime as colunas 2 e 3 de todas as linhas onde encontrar a sequência de caracteres que **termina** em **car** na coluna **col**;
- $ **cat arquivo.dat | awk '$col~/^inicio$/ , $col~/^fim$/ {print $0}'** $\longrightarrow$ Imprime as linhas entre a linha que contém a palavra início e a que contém fim.

Verificar o tamanho de uma coluna ou da linha toda

Verificar o tamanho de uma coluna ou da linha toda

É possível também exibir ou então filtrar as linhas e colunas pelo número de caracteres usando a função lenght().

Verificar o tamanho de uma coluna ou da linha toda

É possível também exibir ou então filtrar as linhas e colunas pelo número de caracteres usando a função lenght().

- $ cat arquivo.dat | awk 'length($1)>5 {print $1, $3}'

Verificar o tamanho de uma coluna ou da linha toda

É possível também exibir ou então filtrar as linhas e colunas pelo número de caracteres usando a função lenght().

- $ cat arquivo.dat | awk 'length($1)>5 {print $1, $3}'
- $ cat arquivo.dat | awk '{print $0, length($0)}'

Verificar o tamanho de uma coluna ou da linha toda

É possível também exibir ou então filtrar as linhas e colunas pelo número de caracteres usando a função lenght().

- $ cat arquivo.dat | awk 'length($1)>5 {print $1, $3}'
- $ cat arquivo.dat | awk '{print $0, length($0)}'
- $ cat arquivo.dat | awk 'length($0)>30 && length($2)<5 {print $0, length($0)}'

Verificar o tamanho de uma coluna ou da linha toda

É possível também exibir ou então filtrar as linhas e colunas pelo número de caracteres usando a função lenght().

- $ cat arquivo.dat | awk 'length($1)>5 {print $1, $3}'
- $ cat arquivo.dat | awk '{print $0, length($0)}'
- $ cat arquivo.dat | awk 'length($0)>30 && length($2)<5 {print $0, length($0)}'
- $ cat arquivo.dat | awk '{print length($1), length($3)}'

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

- $ cat arquivo.dat | awk 'sub(/nome/,"nome2",$1) {print $0}'

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

- $ cat arquivo.dat | awk 'sub(/nome/,"nome2",$1) {print $0}'
- $ cat arquivo.dat | awk 'sub("nome","nome2",$1) {print $0}'

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

- $ cat arquivo.dat | awk 'sub(/nome/,"nome2",$1) {print $0}'
- $ cat arquivo.dat | awk 'sub("nome","nome2",$1) {print $0}'

Caixa alta e baixa

Podemos forçar todas as letras a serem maiúsculas ou minúsculas, tanto de uma coluna específica como da linha toda.

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

- $ cat arquivo.dat | awk 'sub(/nome/,"nome2",$1) {print $0}'
- $ cat arquivo.dat | awk 'sub("nome","nome2",$1) {print $0}'

Caixa alta e baixa

Podemos forçar todas as letras a serem maiúsculas ou minúsculas, tanto de uma coluna específica como da linha toda.

- $ cat arquivo.dat | awk '{print toupper($0)}'

Opções importantes

Substituir palavra ou expressão

Substitui nome por nome2 na coluna 1. Caso a coluna não seja especificada, a troca é feita em todas as colunas ($0);

- $ cat arquivo.dat | awk 'sub(/nome/,"nome2",$1) {print $0}'
- $ cat arquivo.dat | awk 'sub("nome","nome2",$1) {print $0}'

Caixa alta e baixa

Podemos forçar todas as letras a serem maiúsculas ou minúsculas, tanto de uma coluna específica como da linha toda.

- $ cat arquivo.dat | awk '{print toupper($0)}'
- $ cat arquivo.dat | awk '{print $0, $1, tolower($2), $3, $4 }'

Opções importantes

Limitar caracteres

É possível limitar o número de caracteres exibidos de uma coluna ou mesmo da linha inteira, especificado de qual caracter começar a exibição e em qual terminar. Tal função costuma ser mais usualmente aplicada a certas colunas do arquivo e não no arquivo todo.

Opções importantes

Limitar caracteres

É possível limitar o número de caracteres exibidos de uma coluna ou mesmo da linha inteira, especificado de qual caracter começar a exibição e em qual terminar. Tal função costuma ser mais usualmente aplicada a certas colunas do arquivo e não no arquivo todo.

```
Clube P J V E D GP GC SG %
Manchester_City 100 38 32 4 2 106 27 79 87.7
Manchester_United 81 38 25 6 7 68 28 40 71.1
Tottenham 77 38 23 8 7 74 36 38 67.5
Liverpool 75 38 21 12 5 84 38 46 65.8
Chelsea 70 38 21 7 10 62 38 24 61.4
```

Opções importantes

Limitar caracteres

É possível limitar o número de caracteres exibidos de uma coluna ou mesmo da linha inteira, especificado de qual caracter começar a exibição e em qual terminar. Tal função costuma ser mais usualmente aplicada a certas colunas do arquivo e não no arquivo todo.

Clube P J V E D GP GC SG %
Manchester_City 100 38 32 4 2 106 27 79 87.7
Manchester_United 81 38 25 6 7 68 28 40 71.1
Tottenham 77 38 23 8 7 74 36 38 67.5
Liverpool 75 38 21 12 5 84 38 46 65.8
Chelsea 70 38 21 7 10 62 38 24 61.4

- $ cat premier_league.dat | awk '$1=substr($1,1,3) {print $0}'

Opções importantes - Limitar caracteres

```
Clu P J V E D GP GC SG %
Man 100 38 32 4 2 106 27 79 87.7
Man 81 38 25 6 7 68 28 40 71.1
Tot 77 38 23 8 7 74 36 38 67.5
Liv 75 38 21 12 5 84 38 46 65.8
Che 70 38 21 7 10 62 38 24 61.4
Ars 63 38 19 6 13 74 51 23 55.3
Bur 54 38 14 12 12 36 39 -3 47.4
Eve 49 38 13 10 15 44 58 -14 43.0
Lei 47 38 12 11 15 56 60 -4 41.2
New 44 38 12 8 18 39 47 -8 38.6
Cry 44 38 11 11 16 45 55 -10 38.6
Bou 44 38 11 11 16 45 61 -16 38.6
Wes 42 38 10 12 16 48 68 -20 36.8
Wat 41 38 11 8 19 44 64 -20 36.0
```

Exercícios

Atividade catalogo_IRIS.dat

Exercícios

Atividade catalogo_IRIS.dat

1. Conte quantos eventos existem no catálogo com magnitudes entre 6-7, 7-8, 8-9 e > 9;
2. Imprima as linhas entre o EV100 e o EV200;
3. Quantos eventos ocorreram em 2015, 1974? E em abril de 2016?
4. Crie um novo arquivo com todos os eventos que ocorreram no Brasil, imprimindo: long, lat, magnitude e prof;
5. Imprima um novo arquivo para os eventos acima de 7.5 e outro para magnitudes entre 5 e 6, imprimindo: long, lat, magnitude e prof;
6. Digite no terminal ./mapa.sh arquivo, para visualizar os eventos em mapa, para cada arquivo de dados plotado será criado um mapa com o nome: arquivo_events.pdf.